Efficient Model Checking with Semantically-Equivalent Models for *vGOAL*

Extended Abstract

Yi Yang
imec-DistriNet, KU Leuven
Leuven, Belgium
yi.yang@kuleuven.be

Tom Holvoet
imec-DistriNet, KU Leuven
Leuven, Belgium
yi.yang@kuleuven.be

ABSTRACT

Model checking offers a powerful approach to ensuring safety and reliability in autonomous systems. However, existing model-checking approaches for agent programming languages (APLs) face challenges in equivalent semantic mapping, efficient model generation, and integration with high-performance model checkers. We present a computation tree logic (CTL) model-checking framework for *vGOAL*, where both the interpreter and model-checking framework share the same state update implementations. Our framework establishes semantically equivalent models of *vGOAL* programs, implements efficient state space generation, and integrates with the NuSMV model checker. Through a case study of an autonomous logistic system with up to three robots, we demonstrate significant improvements in model-checking efficiency, enabling verification of complex autonomous systems.

KEYWORDS

CTL Model Checking,*vGOAL*, Autonomous Decision-Making

ACM Reference Format:
Yi Yang and Tom Holvoet. 2025. Efficient Model Checking with Semantically-Equivalent Models for *vGOAL*: Extended Abstract. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 3 pages.

1 INTRODUCTION

As autonomous systems become more prevalent, APLs like AgentSpeak [5], Gwendolen [11], and GOAL [17] have emerged as essential tools for programming autonomous decisions. These languages are designed to model autonomous decision-making by enabling agents to perceive their environment and make decisions based on rules. Formal verification through model checking offers a promising approach to ensure their reliability [22, 25], and numerous efforts have been dedicated to model checking agent programs [2–4, 8, 12–14, 16, 18, 21, 24, 26, 29]. Existing methods face significant challenges in semantic equivalence, model generation efficiency, and scalability [3, 15, 16, 21].

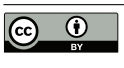
Modern advancements in model checkers enable handling state spaces with hundreds of millions of states [1, 6, 7, 9, 19], making them suitable for verifying complex autonomous systems. However, applying these to APLs involves difficulties such as semantically-equivalent model generation, inefficiency of interpreters, and limited scalability for multi-agent settings. Current interpreter-based frameworks like MCAPL [10] and IMC [20, 21] have been applied to languages such as Gwendolen, GOAL, and AgentSpeak [18] but show limitations like inefficiency [16]. Alternative non-interpreter-based methods [3] suffer from complicated semantic mapping from agent programs to the input format for a model checker, like Promela [23]. Our recent work [27, 29] ensures semantic equivalence for GOAL programs but is constrained to single-agent systems.

To address these challenges, we develop a CTL model-checking framework for *vGOAL* [28], a variant of GOAL designed for verifiable autonomous decision-making. Our framework makes three key contributions: (1) establishing semantically-equivalent models for *vGOAL* programs, (2) developing an efficient state space generation algorithm, and (3) integrating NuSMV for efficient CTL model-checking. To demonstrate the practical application and scalability of our framework, we conduct a case study involving a logistics transportation system with multiple autonomous robots. The results demonstrate substantial improvements in verification efficiency and scalability, establishing a foundation for model-checking complex multi-agent autonomous systems programmed in *vGOAL*.

2 CTL MODEL-CHECKING FRAMEWORK

The CTL model-checking framework for *vGOAL* is designed to enable the efficient verification of the semantically-equivalent models of *vGOAL* programs. The *vGOAL* model-checking framework is an interpreter-based model-checking framework, and it shares the same state update implementationits with its interpreter [30], enabling semantically-equivalent model generation. As illustrated in Figure 1, the framework consists of three components: non-deterministic modeling, NuSMV encoding, and NuSMV verification. Our primary focus is on the first two components—non-deterministic modeling and NuSMV encoding. Non-deterministic modeling encompasses both the generation of semantically-equivalent models and the implementation of efficient model generation techniques.

Non-Deterministic Modeling. The semantically-equivalent model generation begins by constructing a transition system from a given *vGOAL* program. The transition system captures all possible states, transitions, and terminal states of the program. Each state is iteratively expanded to explore all agent transitions, with terminal states identified when agents have completed their goals. By sharing state update implementations with the *vGOAL* interpreter, the framework ensures the generated model accurately reflects the



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

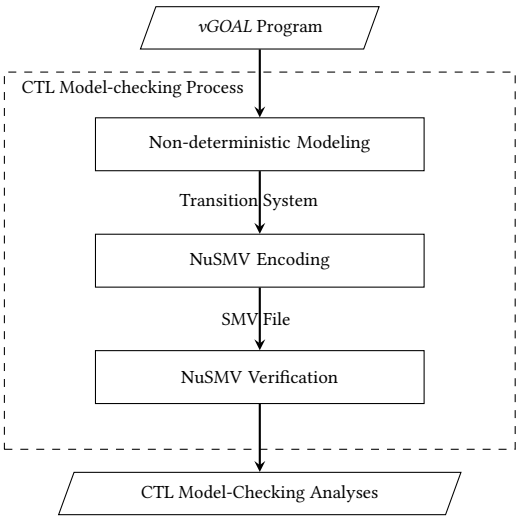


Figure 1: The Overview of CTL Model-Checking Framework for *vGOAL*.

exact execution behavior of the original program. This equivalence can be formally established through the existence of a bisimulation relation between the transition system and all possible executions of the *vGOAL* program. Additionally, atomic proposition labeling is employed to represent properties based on state attributes.

To enhance efficiency, optimizations are applied during state exploration. A caching mechanism prevents redundant computation for revisited states, and deterministic updates efficiently handle instances where agents switch focus between goals. Both techniques contribute to significant computational savings without compromising semantic correctness. An example of a generated transition system illustrates the iterative state expansion and the way agent transitions are modeled, showcasing the straightforward representation of non-deterministic agent behaviors. The framework implements an optimization for state exploration in the form of an improved expansion function. This function incorporates a caching mechanism to store previously computed transitions, significantly reducing redundant computations when states are revisited. For scenarios in which agents dynamically switch focus between goals, the approach efficiently updates transitions to reflect the altered goal prioritization. The improved function produces a transition system that is identical to the one generated by the original method, ensuring full correctness while delivering better computational efficiency.

NuSMV Encoding. Once the transition system is generated, it is encoded into a NuSMV-compatible format. This encoding process begins by preprocessing the system's properties, including safety, error, and termination conditions defined in the *vGOAL* program. These properties are classified and mapped to the states of the transition system. The encoding process converts each component of the transition system—states, transitions, terminal states, atomic propositions, and associated properties—into their respective representations in an SMV file. This automated encoding includes CTL properties for safety (to ensure that no undesirable states are reached) and liveness (to check the existence of paths leading to terminal states). For example, for a transition system representing agent behaviors, the encoding might specify that the agent is always in a safe or valid state and check the existence of error-free paths to a terminal state. These CTL properties enable NuSMV to validate the program's correctness and reliability, ensuring the autonomous system behaves as designed.

NuSMV Verification. NuSMV takes the encoded SMV file as input and automatically performs the CTL model-checking analysis, and it produces the CTL model-checking analysis as the output, indicating whether the specified CTL properties are satisfied or not. These results are crucial for validating the reliability and correctness of the *vGOAL* program, ensuring that the autonomous system behaves as intended.

3 EMPIRICAL ANALYSES

We evaluated our CTL model-checking framework through a case study of an autonomous logistics system, comparing the original framework against the improved version that incorporates efficient model generation algorithms. The experiments were performed under three configurations involving one, two, and three agents, each capable of managing multiple delivery goals.

Our analysis revealed that both the number of goals and agents correlate positively with the state space size, with agent count having a more significant impact. Notably, the state space converges to a finite value regardless of further increases in goals or agents. The improved framework consistently outperforms the original version across all test scenarios, with modeling time dominating the overall execution time in both versions.

The performance gap between the two versions becomes more significant as system complexity increases. This superior performance stems from reducing the time complexity from $O(goals \times states)$ to $O(states)$. This optimization proves particularly valuable for complex scenarios involving multiple goals and agents, where the original framework's performance degrades significantly.

4 CONCLUSIONS AND FUTURE WORK

This paper presents a comprehensive CTL model-checking framework for *vGOAL* that effectively addresses three major challenges in APL model-checking: semantically-equivalent mapping, efficient model generation, and integration with efficient model checkers. Through empirical analysis of 30 scenarios, we demonstrate that our efficient model generation algorithm significantly reduces time complexity while efficiently verifying complex autonomous systems, with the notable observation that state space converges to a constant number as goals increase. Future work will explore the theoretical underpinnings of state space convergence in *vGOAL* programs, aiming to simplify model generation by identifying the point of convergence and enhancing the verification of multi-goal autonomous systems.

ACKNOWLEDGEMENTS

This research is partially funded by the Research Fund KU Leuven.

REFERENCES

[1] Roman Andriushchenko, Alexander Bork, Carlos E Budde, Milan Češka, Kush Grover, Ernst Moritz Hahn, Arnd Hartmanns, Bryant Israelsen, Nils Jansen, Joshua Jeppson, et al. 2024. Tools at the Frontiers of Quantitative Verification. *arXiv preprint arXiv:2405.13583* (2024).

[2] Rafael H Bordini, Louise A Dennis, Berndt Farwer, and Michael Fisher. 2008. Automated verification of multi-agent programs. In *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 69–78.

[3] Rafael H Bordini, Michael Fisher, Carmen Pardavila, and Michael Wooldridge. 2003. Model checking AgentSpeak. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. 409–416.

[4] Rafael H Bordini, Michael Fisher, Willem Visser, and Michael Wooldridge. 2004. Verifiable multi-agent programs. In *Programming Multi-Agent Systems: First International Workshop, PROMAS 2003, Melbourne, Australia, July 15, 2003, Selected Revised and Invited papers 1*. Springer, 72–89.

[5] Rafael H Bordini and Jomi F Hübner. 2005. BDI agent programming in AgentSpeak using Jason. In *International workshop on computational logic in multi-agent systems*. Springer, 143–164.

[6] Carlos E Budde, Arnd Hartmanns, Michaela Klauck, Jan Křetínskỳ, David Parker, Tim Quatmann, Andrea Turrini, and Zhen Zhang. 2020. On Correctness, Precision, and Performance in Quantitative Verification. In *International Symposium on Leveraging Applications of Formal Methods*. Springer, 216–241.

[7] Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, Marco Roveri, et al. 1999. NuSMV: A new symbolic model verifier. In *CAV*, Vol. 99. Citeseer, 495–499.

[8] Mehdi Dastani, Nick AM Tinnemeier, and John-Jules Ch Meyer. 2009. A programming language for normative multi-agent systems. In *Handbook of Research on Multi-Agent Systems: semantics and dynamics of organizational models*. IGI Global, 397–417.

[9] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. 2017. A Storm is coming: A modern probabilistic model checker. In *International Conference on Computer Aided Verification*. Springer, 592–600.

[10] Louise A Dennis. 2018. The MCAPL framework including the agent infrastructure layer and agent java pathfinder. *The Journal of Open Source Software* (2018).

[11] Louise A Dennis and Berndt Farwer. 2008. Gwendolen: A BDI language for verifiable agents. In *Proceedings of the AISB 2008 Symposium on Logic and the Simulation of Interaction and Reasoning, Society for the Study of Artificial Intelligence and Simulation of Behaviour*. Citeseer, 16–23.

[12] Louise A Dennis and Michael Fisher. 2008. Programming verifiable heterogeneous agent systems. In *International Workshop on Programming Multi-Agent Systems*. Springer, 40–55.

[13] Louise A Dennis and Michael Fisher. 2020. Verifiable self-aware agent-based autonomous systems. *Proc. IEEE* 108, 7 (2020), 1011–1026.

[14] Louise A Dennis, Michael Fisher, Nicholas K Lincoln, Alexei Lisitsa, and Sandor M Veres. 2016. Practical verification of decision-making in agent-based autonomous systems. *Automated Software Engineering* 23 (2016), 305–359.

[15] Louise A Dennis, Michael Fisher, and Matt Webster. 2018. Two-stage agent program verification. *Journal of Logic and Computation* 28, 3 (2018), 499–523.

[16] Louise A Dennis, Michael Fisher, Matthew P Webster, and Rafael H Bordini. 2012. Model checking agent programming languages. *Automated software engineering* 19, 1 (2012), 5–63.

[17] Koen V Hindriks. 2009. Programming rational agents in GOAL. In *Multi-agent programming*. Springer, Berlin, Heidelberg, 119–157.

[18] Koen V Hindriks, Frank S De Boer, Wiebe Van der Hoek, and John-Jules Ch Meyer. 1999. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems* 2 (1999), 357–401.

[19] Gerard J. Holzmann. 1997. The model checker SPIN. *IEEE Transactions on software engineering* 23, 5 (1997), 279–295.

[20] S.S.T.Q. Jongmans. 2010. Model checking GOAL agents.

[21] Sung-Shik TQ Jongmans, Koen V Hindriks, and M Birna Van Riemsdijk. 2010. Model checking agent programs by using the program interpreter. In *Computational Logic in Multi-Agent Systems: 11th International Workshop, CLIMA XI, Lisbon, Portugal, August 16-17, 2010. Proceedings 11*. Springer, 219–237.

[22] Matt Luckcuck, Marie Farrell, Louise A Dennis, Clare Dixon, and Michael Fisher. 2019. Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–41.

[23] Erich Mikk, Yassine Lakhnech, Michael Siegel, and Gerard J Holzmann. 1998. Implementing statecharts in PROMELA/SPIN. In *Proceedings. 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques*. IEEE, 90–101.

[24] M Birna Van Riemsdijk, Frank S De Boer, Mehdi Dastani, and John-Jules Ch Meyer. 2006. Prototyping 3APL in the Maude term rewriting language. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. 1279–1281.

[25] Gerhard Weiss. 2013. *Multiagent Systems*. The MIT Press, Cambridge. 462 pages.

[26] Michael Winikoff. 2007. Implementing commitment-based interactions. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. 1–8.

[27] Yi Yang and Tom Holvoet. 2022. Making Model Checking Feasible for GOAL. In *10th International Workshop on Engineering Multi-Agent Systems*.

[28] Yi Yang and Tom Holvoet. 2023. *vGOAL*: a GOAL-based Specification Language for Safe Autonomous Decision-Making. In *Engineering Multi-Agent Systems: 11th International Workshop, EMAS 2023, London, UK, 29-30 May 2023, Revised Selected Papers*.

[29] Yi Yang and Tom Holvoet. 2023. Making model checking feasible for GOAL. *Annals of Mathematics and Artificial Intelligence* (2023). https://doi.org/10.1007/s10472-023-09898-3

[30] Yi Yang and Tom Holvoet. 2023. Safe Autonomous Decision-Making with vGOAL. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*. Guimarães, Portugal.